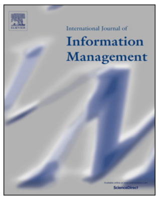
International Journal of Information Management xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

International Journal of Information Management

journal homepage: www.elsevier.com/locate/ijinfomgt



Autonomic machine learning platform

Keon Myung Lee[a], Jaesoo Yoo[b], Sang-Wook Kim[c], Jee-Hyong Lee[d], Jiman Hong[e,*]

[a] Dept of Computer Science, Chungbuk National University, Cheongju, Republic of Korea
[b] School of Information and Communication Engineering, Chungbuk National University, Cheongju, Republic of Korea
[c] Department of Computer Science, Hanyang University, Seoul, Republic of Korea
[d] Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, Republic of Korea
[e] School of Computer Science and Engineering, Soongsil University, Seoul, Republic of Korea

ARTICLE INFO

Keywords:
Autonomic machine learning platform
Autonomic level
Machine learning
Smart City

ABSTRACT

Acquiring information properly through machine learning requires familiarity with the available algorithms and understanding how they work and how to address the given problem in the best possible way. However, even for machine-learning experts in specific industrial fields, in order to predict and acquire information properly in different industrial fields, it is necessary to attempt several instances of trial and error to succeed with the application of machine learning. For non-experts, it is much more difficult to make accurate predictions through machine learning.

In this paper, we propose an autonomic machine learning platform which provides the decision factors to be made during the developing of machine learning applications. In the proposed autonomic machine learning platform, machine learning processes are automated based on the specification of autonomic levels. This autonomic machine learning platform can be used to derive a high-quality learning result by minimizing experts' interventions and reducing the number of design selections that require expert knowledge and intuition. We also demonstrate that the proposed autonomic machine learning platform is suitable for smart cities which typically require considerable amounts of security sensitive information.

1. Introduction

Machine learning has radically advanced and has advanced into many areas of our lives over the past several years. A variety of machine learning algorithms that are very effective at extracting knowledge from big data in the form of patterns or models have been developed. The adoption of machine learning algorithms can increase productivity, quality, and profit levels by greatly reducing costs and labor requirements in academia as well as industry. Recent great progress in machine learning technology is due to the invention of large new machine learning algorithms such as deep learning which is a subset of machine learning, has been one of the essential enablers for the renewed AI success (Duan, Edwards, & Dwivedi, 2019) and increased availability of huge data and computing resources. These are being applied to various business domains and accordingly increasing the size of the machine learning market (Bergstra & Bengio, 2012; Bergstra, Bardenet, Bengio, & Kégl, 2011; Brochu, Cora, & De Freitas, 2010; Cai et al., 2014; Kang, Lee, & Lee, 2015; Larochelle, Erhan, Courville, Bergstra, & Bengio, 2017; Lee, Lam, Pedarsani, Papailiopoulos, & Ramchandran, 2018; Thornton, Hutter, Hoos, & Leyton-Brown, 2013).

Over the last few years, a variety of machine learning platforms which support various machine learning algorithms to users have been developed, including well-known machine learning libraries and frameworks such as the Scikit-Learn Python libraries, Google's TensorFlow, and Microsoft Research's CNTK. As a result, many non-experts have conducted various studies, applied machine learning to various applications, and extracted various features. As a result, anyone who knows how to use a computer can find it easy to use a machine learning tool and/or framework to solve numerous problems and consider techniques that can be applied to any situation (Branscombe, 2018).

Despite the many studies conducted to build machine learning platforms capable of learning to make intelligent decisions, there have historically been many barriers to use machine learning efficiently because machine learning is not magic for software developers (Chandrasegaran, 2017). One of the major barriers is that different machine learning algorithms should be used depending on the size, quality, and characteristics of the data. Another barrier is that different

* Corresponding author at: 369 Sangdo-ro, Dongjak-gu, Seoul, 06978, Republic of Korea.
E-mail addresses: kmlee@cbnu.ac.kr (K.M. Lee), yjs@chungbuk.ac.kr (J. Yoo), wook@agape.hanyang.ac.kr (S.-W. Kim), john@skku.edu (J.-H. Lee), jiman@ssu.ac.kr (J. Hong).

https://doi.org/10.1016/j.ijinfomgt.2019.07.003
Received 31 October 2018; Received in revised form 27 June 2019; Accepted 1 July 2019

machine learning algorithms should be used depending on the mathematical algorithms as instructions for machine learning computer systems. Finally, different machine learning algorithms are used necessary depending on the execution time.

Selecting the right algorithm, constructing suitable input features by experimentation with different parameters, and testing and cross-validation robustly are overly challenging for non-experts. Most non-experts do not know which machine learning algorithms work optimally before trying them out. Even experts often apply various machine learning algorithms when they must deal with data not frequently encountered in machine learning.

Machine learning is difficult to apply successfully to a specific problem for several reasons. First, the appropriate machine learning task should be identified as a problem. Tasks include classification, regression, clustering, recommendation, density estimation, dimensionality reduction, feature representation, and others. Second, it is of paramount importance to use proper features by selecting or extracting features from data sets with various attributes. Third, there are many candidate machine learning algorithms for a specific task, each of which has some advantages or disadvantages over others. Fourth, most machine learning algorithms have certain hyperparameters, which are parameters that influence the algorithm's behavior. The existence of hyperparameters requires expert intervention to select their proper values. In fact, it is impossible to examine combinations of hyperparameter values by executing machine learning algorithms without the insight of experts and a generate-and-test approach.

Recent progress in machine learning, especially in deep learning and probabilistic graphical models, take a considerable amount of computation due to model complexity in these cases. Machine learning algorithms with many hyperparameters require considerable computing resources for training and inference purposes (Bergstra & Bengio, 2012; Bergstra et al., 2011; Brochu et al., 2010; Thornton et al., 2013). Therefore, it is impossible to compute these numerous computations without external computing resources such as cloud computing systems or distributed computing systems (Kang et al., 2015; Larochelle et al., 2017; Lee et al., 2018).

There are a great many excellent open-source machine learning tools and frameworks, such as TensorFlow, Keras, Scikit-learn, Microsoft Cognitive Toolkit, Theano, Caffe, PyTorch, and Accord.NET. However, most industry, business, and government domains have difficulty when applying machine learning techniques to their various applications without experts.

There is an increasing demand for experts and skilled engineers in the areas of machine learning across all industries, but well-trained machine learning experts represent a rare resource. Therefore, there is an increasing level of demand for an efficient machine learning platform that enables not only non-experts who have no knowledge or experience with regard to machine learning but also that enables experts to use machine learning algorithms and various machine learning tools more easily.

In order to obtain meaningful information through machine learning and to provide various services to large-scale applications such as smart cities, smart factories, and smart grid systems, which continuously produce various types of big data, more experts are needed relative to those needed for other applications. In particular, smart cities, which contain a wealth of sensitive data, require additional security systems to ensure the security of sensitive data. This ultimately increases the economic burden associated with securing sensitive data. Therefore, a new machine learning platform is needed to reduce these types of economic burdens while also minimizing the number of experts required for machine learning.

In this paper, we propose an autonomic machine learning platform. The proposed platform is based on five autonomic levels. These levels are based on the categorizing the machine learning processes into several development steps and defining each development step according to the degree of expert intervention. The determining factor for each level is whether it realizes an effective and efficient machine learning application.

We also present the requirements of an autonomic machine learning platform which helps non-expert researchers and developers build machine learning applications. Finally, we demonstrate how the proposed platform can be suitable for a smart city with sensitive big data.

The remainder of the paper is organized as follows: Section 2 presents work related to machine learning automation. Section 3 presents the autonomic machine learning level. Section 4 presents the design factors of autonomic machine learning and the requirements of an autonomic learning platform. Section 5 presents the architecture of the proposed autonomic machine learning platform and the functionalities of the component modules in the platform. This section also demonstrates how the proposed autonomic machine learning platform is particularly suitable for a smart city. Section 6 presents the theoretical contributions and limitations of the research. Finally, Section 7 concludes the paper.

2. Related work

In this section, we present an overview of existing literature on autonomic machine learning. There have been few breakthrough attempts to study autonomic machine learning platforms or self-directed learning. Standardization also specifies the computing environment required for the international standardization of high-performance machine-learning frameworks and focuses only on software development. The autonomic extraction of features as an autonomic learning method for machine learning models has been studied, and there has been some effort to determine hyperparameters automatically in machine algorithm algorithms.

Other studies of autonomic machine learning without expert intervention in the areas of model learning and hyperparameter selection have also been conducted (Feurer et al., 2015; Moore, 2018). With Cloud AutoML, Google now provides a commercial service that allows developers with limited knowledge of machine learning to develop a machine learning model using Google's environment (Google Cloud AutoML BETA, 2018). However, currently most autonomic machine learning focuses on the choices of hyperparameter values.

Methods for determining hyperparameters include grid searches, random searches, Powell's method, the Nelder-Mead method, tree-based Parzen estimation (HyperOpt), sequential model-based algorithm setting (Auto-Weka), and Gaussian process-based models. The grid search method generates candidate hyperparameter combinations, with each hyperparameter equally spaced in its domain and allows the selection of the combination which results in the best performance by executing a corresponding machine learning algorithm for each candidate (Larochelle et al., 2017).

The random search method generates candidate hyperparameter combinations randomly to provide empirically better odds of finding better hyperparameters within the given computation budget (Bergstra & Bengio, 2012). The sequential model-based algorithm configuration method selects a hyperparameter combination based on a model rather than uniformly at random (Bergstra et al., 2011). The tree-structured Parzen estimator sequentially constructs a model which approximates the performance of hyperparameters based on historical measurements. This method then chooses new hyperparameters to test based on the constructed model (Bergstra et al., 2013). The Gaussian process-based method uses a Gaussian process as a surrogate for hyperparameter distribution and updates the Gaussian process by combining sampling results and a prior distribution (Brochu et al., 2010).

Typical machine learning tools include Auto-sklearn, Auto-Weka, TPOT, and Google's Vizier. Auto-sklearn uses a sequential model-based algorithm configuration (SMAC) to select algorithms and hyperparameters (ML Freiburg, 2018). The algorithm configuration is based on a Bayesian optimization random pattern that repeats the process of the learning of a model by selecting a new hyperparameter value while creating a probability model representing the relationship between the hyperparameters and the machine learning model performance. Auto-Weka is a machine learning tool based on Weka; it uses the Bayesian optimization technique SMAC as well as Auto-sklearn for algorithm selection and to determine the hyperparameter (Kotthoff, 2018).

Tree-based pipeline optimization (TPOT) uses Scikit-learn to study various combinations of algorithms and hyperparameters using genetic programming (Olson & Moore, 2016). The H2O driverless AI provides a web-based user interface. The results are provided in a chart with annotation. Google Vizier uses the implementation of a black box optimization service which supports random searches, grid searches, and Gaussian-process-based Bayesian optimization methods to determine hyperparameters in Google's Cloud AutoML (Golovin et al., 2017).

Due to the considerable levels of computational demand, there have been various works on distributed processing and parallel processing in machine learning. OptiML, GraphLab, SystemML, SimSQL, and MLBase are related machine learning platforms that provide programming and runtime support (Cai et al., 2014). OptiML was developed to allow a machine learning practitioner to write code in a MATLAB-like declarative manner, with the code able to run on various hardware platforms, such as a multi-core CPU, a GPU, clusters of computing nodes, and other specialized accelerators.

GraphLab is a graph-based, distributed computation framework which uses graph-parallel abstraction for sparse iterative graph algorithms and works in a pull-based and asynchronous manner. SystemML is a machine learning framework in which a declarative machine learning language is used to describe machine learning algorithms and where the codes expressed in the machine learning language are compiled and optimized into hybrid runtime plans of multi-threaded, in-memory operations in a single node or distributed map-reduce or Apache Spark operations on the cluster of nodes. SimSQL is a SQL-based platform which runs on top of Hadoop and which is designed to support scalable Bayesian machine learning. MLBase is a machine learning framework based on Apache Spark for lower-level data processing. It works on the Hadoop platform. Google's TensorFlow graphs machine learning work and allows the graph's work to be run via the GPU and through remote processes.

Giraph, Apache Spark, and DryadLinq are some of the frameworks with which machine learning tasks are programmed and executed, though they were not designed only for machine learning tasks (Sparks et al., 2013). Giraph is a graph-based distributing computing framework in which models are push-based and synchronous. Apache Spark is a cluster computing framework for large-scale data analytics which utilizes resilient distributed datasets (RDDs). It allows in-memory computation and provides fault-tolerance by managing data lineage. DryadLinq is a distributed computing framework which uses the distributed execution engine Dryad and the. Net-language-integrated query LINQ and allows the simple development of machine learning algorithms as well as other distributed computing applications.

3. Autonomic machine learning level

In this section, first we define autonomic machine learning based on minimizing expert intervention. We also define the autonomic levels based on the development factors of the machine learning process.

3.1. Autonomic machine learning

Similar to the concept of autonomic computing, which refers to a computing framework to manage, configure, and optimize the assets of all systems while minimizing expert intervention (Autonomic Computing Strategy Perspectives, 2018), autonomic machine learning refers to autonomic machine learning with minimal expert intervention. Therefore, autonomic machine learning can be defined as the selection of an appropriate machine learning model and algorithm to achieve the desired result while autonomously detecting the characteristics of the data.

The performance of machine learning generally can be classified in terms of the quality of the machine learning results, the consumption of computer resources, and by how much expert intervention is minimized. Quality performance of the machine learning result means that how the model learned by applying a machine learning algorithm is superior in terms of accuracy. The performance with regard to the consumption of computer resources means that a machine learning model and algorithm are superior in terms of resource consumption metrics, such as the CPU time, the amount of memory, and the amount of disk space used.

Finally, performance with reference to minimizing expert intervention means that machine learning can be applied to an application with minimal expert interventions needed due to issues such as difficulties in determining the type of the machine learning algorithm, the model setting, or the hyperparameters.

However, minimizing expert intervention in machine learning is challenging to implement easily with current computing technologies. In order to apply machine learning, experts or developers must understand the relevant algorithms when selecting an appropriate one, and they must test a number of combinations of hyperparameters, each of which determines the operating characteristics of the algorithm, to find the best set. This series of processes makes it difficult to minimize expert intervention. It is more difficult for the machine to determine the hyperparameters on its own because the machine learning algorithm requiring a large amount of computations should be repeatedly performed to determine the proper hyperparameters. A possible solution is to define the development factors of the machine learning process as a general rule and to categorize these factors into a specific step so that the system can be implemented at the machine learning process level without direct human intervention. In addition, if there is a system capable of automatically selecting not only hyperparameters such as the learning model type, loss function coefficient, and learning rate but also a machine learning algorithm suitable for the problem while utilizing external computing resources such as cloud computing and distributed computing systems, the proposed autonomic machine learning will work well.

3.2. Autonomic levels

In order to develop machine learning applications which work by learning target data, machine learning experts generally undertake the processes of selecting the attributes of the input data, tuning the hyperparameters, and selecting the machine learning technique and learning task. If we categorize these processes into the development steps of machine learning and if the process of each step can be performed without expert intervention, each step can then be defined as a level of autonomic machine learning. Therefore, we define five levels of autonomic machine learning referring to as the degree of expert intervention based on the development steps of machine learning. These are shown in Table 1.

Table 1
Autonomic Levels of Machine Learning.

Autonomic Level	Degree of Expert Intervention(Requirements)
4	Given target data
3	Given target data, Machine learning task
2	Given target data, Machine learning task, Machine learning technique(algorithm)
1	Given target data, Machine learning task, Machine learning technique(algorithm), Hyperparameter
0	Given target data, Machine learning task, Machine learning technique(algorithm), Hyperparameter, Input data's attribute

At autonomic level 0, all steps of the above-mentioned processes of machine learning are assumed to be provided by experts or developers. Autonomic level 0 is the current level of machine learning development, where no autonomic machine learning functionality is provided.

At autonomic level 1, experts or developers are freed from having to select proper input attributes during the development of machine learning applications. In machine learning applications, it is important to determine the attributes relevant to the output attribute, especially in supervised learning tasks. For each type of machine learning task, there are several techniques that can be applied for feature selection and extraction. Autonomic level 1 should provide proper functionalities for extracting relevant features with no expert intervention.

At autonomic level 2, the functionalities for determining the hyperparameters for a given algorithm should be provided, matching the capability of autonomic level 1. The hyperparameters are among the major factors to maximize the performance of a specific machine learning algorithm on a validation data set because they directly control the behavior of the machine learning algorithm. Setting and tuning the hyperparameters is usually time-consuming because it is somewhat a trial-and-test task with no golden law for setting and tuning the best hyperparameters despite the fact that this task has been actively studied in relation to machine learning algorithms. Therefore, setting and tuning the hyperparameters is usually done by experts with a wealth of experience and a strong understanding of machine learning algorithm.

At autonomic level 3, only the training data set and its machine learning task are given, and all other functionalities for other machine learning works should be provided. At this level, the machine learning algorithm to use, the hyperparameter settings, and the selection of the relevant input attributes are provided. It is necessary to evaluate candidate machine learning algorithms for a given task and to determine the hyperparameters for each machine learning algorithm under consideration. The brute-force approach is not a good approach when searching for a feasible machine learning model due to the excessive number of possible models.

Strategies to reduce the search space are necessary. One strategy is to use expert knowledge pertaining to the most effective machine learning algorithm in each situation. This knowledge-based approach requires the building of a knowledge base through expert knowledge and literature surveys. However, this strategy is insufficient if used to find a good model.

At autonomic level 4, the machine learning application is developed when only the training data is provided. All other functionalities for other machine learning processes should be provided. At this level, the tasks that can be conducted for a given data set are determined by conducting unsupervised learning on the data set. This form of unsupervised learning is rather broad compared to unsupervised learning commonly referred to in the machine learning literature because autonomic level 4 is concerned with not only supervised tasks but also with unsupervised tasks.

Autonomic level 5 is the ultimate level at which nothing needs to be created by developers. Here, certain patterns are sought from all available factors without any expert or developer intervention.

The higher the level, the less the expert intervention and the more fully autonomic machine learning is supported because each level of autonomic machine learning is defined as a level in a hierarchy that always includes the requirements of the next level below it. Likewise, lower autonomic levels are associated with greater requirements of the machine learning model with more professional intervention required. Hence, the level of autonomic machine learning indicates how much expert intervention is required, and this parameter can be used as a general guideline for how technically advanced the autonomic machine learning is on a case-by-case basis.

Autonomic machine learning is an innovative technique which transforms machine learning without expert experience in the future. With this technique, people do not have to worry about selecting or creating the features of big data and will therefore simply leave it all up to the computer system. It is very difficult for computer systems to learn as comprehensively as experts because the scope of learning is broad; currently developed machine learning methods have limitations related to the subject and scope.

However, though the ideal full autonomic machine learning does not appear to be possible with current technology, it will be possible in the future. Even partial autonomic functionalities can be provided, as computer systems are becoming faster, new machine learning algorithms are being developed, and more data will be learned by a variety of machine learning computer systems. Therefore, autonomic machine learning will be able to grasp the attributes of a given data set, extract the features of data without experts, and will be able to make new predictions independently. A variety of features delivering partial autonomic machine learning will be available, with each level serving as a springboard for full autonomic machine learning in the future.

4. Requirements of autonomic machine learning

In this section, we present the requirements for making autonomic machine learning possible by examining the design choices in machine learning. The design choices are made by experts or developers when they develop machine learning applications.

4.1. Design factors of autonomic machine learning

Despite many available machine learning tools, it takes an expert to use them effectively with an actual problem because there are various factors to be selected during the application of a machine learning algorithm. In the autonomic machine learning model, because the processes of the generation and optimization of data are conducted based on the characteristics of the data rather than on the subjective knowledge of an expert, it takes a considerable amount of time and incurs a high cost to generate various types of data and to process large amounts of data. Therefore, careful selections of various design factors, are necessary, as shown in Fig. 1, to minimize expert intervention and reduce the time and cost to build the autonomic machine learning platform properly. The characteristics of typical design factors considered by machine learning experts are discussed below.

4.1.1. Acquisition of training data and partitioning

Machine learning models are developed from training data. Hence, the acquisition of good training data is of paramount importance, as is collecting data that reflect the problem domain in a cost-effective manner. The collected data are used for training, validation, and testing. This data must be partitioned into smaller collections according to the machine learning algorithm employed and the application context. Factors to consider include the amount of data available, the testing and validation methods, and the budget of the computing resources.

4.1.2. Preprocessing of data

At times, data must be preprocessed because the data may not be clean or well curated. Data cleansing, transformation, and normalization are typical tasks conducted at this stage. Most machine learning algorithms have an independent format for input data, and raw data do not usually follow the format. Data transformation should therefore be performed to convert the data into an amenable format. Data normalization is one of the fundamental steps during the data preparation process. The domains of the attributes in the data can differ from one another. When the degree of a distance or similarity is computed, differences in a range of data attributes may distort the metrics. Normalization serves to standardize the data and make the scale uniform.

4.1.3. Feature extraction

Most machine learning algorithms are strong enough to extract

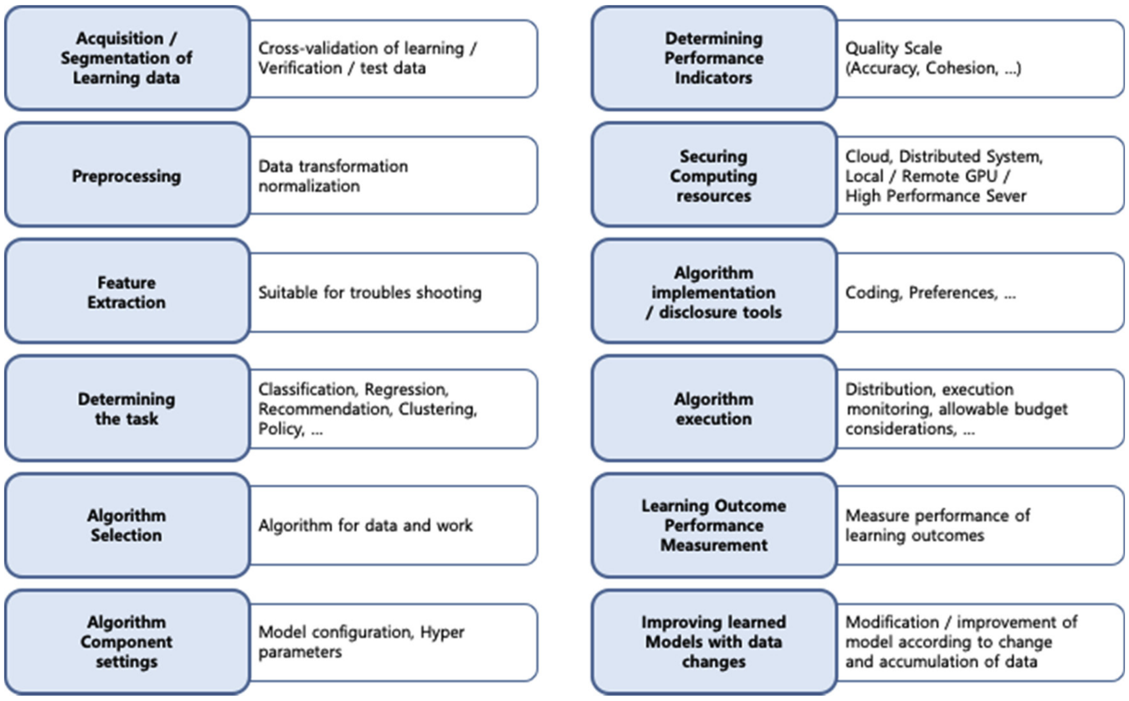

Acquisition / Segmentation of Learning data
Cross-validation of learning / Verification / test data
Preprocessing
Data transformation normalization
Feature Extraction
Suitable for troubles shooting
Determining the task
Classification, Regression, Recommendation, Clustering, Policy, ...
Algorithm Selection
Algorithm for data and work
Algorithm Component settings
Model configuration, Hyper parameters
Determining Performance Indicators
Quality Scale (Accuracy, Cohesion, ...)
Securing Computing resources
Cloud, Distributed System, Local / Remote GPU / High Performance Sever
Algorithm implementation / disclosure tools
Coding, Preferences, ...
Algorithm execution
Distribution, execution monitoring, allowable budget considerations, ...
Learning Outcome Performance Measurement
Measure performance of learning outcomes
Improving learned Models with data changes
Modification / improvement of model according to change and accumulation of data

Fig. 1. Design factors in the machine learning process.

features while learning a model for a designated task. The quality of features has a strong influence on the performance of the final machine learning system. The choice of quality features requires expertise and is occasionally a time-consuming task. Feature extraction techniques include both feature selection and generation methods. Feature selection methods choose certain features from among those available. There are several measures, such as cross-entropy, for evaluating the relevance of the attributes of the task of interest. Feature generation methods produce new features by combining the available attributes. A recently developed deep learning technique allows developers to pay less attention to feature extraction.

4.1.4. *Task decision*

There are various types of machine learning tasks, such as classification, regression, recommendation, clustering, and policy construction. Classification refers to the mapping of data into a pre-defined class. Regression can be used to map the input data into output data in a numerical domain. Recommendation refers to the suggestion of related items based on the history of user activities. The clustering of groups creates subgroups based on similarity. Policy construction refers to the selection of actions proper for the target situation.

4.1.5. *Machine learning algorithm selection*

Machine learning approaches are broadly categorized into supervised learning, unsupervised learning, and reinforcement learning. In the machine learning approaches, there are many choices of machine learning algorithms, each of which has its own strengths and weaknesses. It takes an expert to choose the proper machine learning algorithm for a given problem domain.

4.1.6. *Machine learning algorithm configuration*

Most algorithms require some configuration efforts, such as model configuration and hyperparameter setting. Some machine learning algorithms specify a general strategy of problem-solving, thus requiring developers to determine the details of the algorithms. In a genetic algorithm, chromosome coding of candidate solutions and genetic operators are the main components determined by developers for problem handling. In deep learning and neural networks, the network configuration critically affects the performance of the machine learning application. Hyperparameters are parameters for developers to set based on their own experiences and insights.

4.1.7. *Performance measure selection*

Machine learning algorithms attempt to improve machine learning models with respect to certain performance measures. Machine learning algorithms refine their performance measure candidates. Developers must choose the performance measure to use for a machine learning application and determine the criteria for accepting the developed model.

4.1.8. *Computing resource acquirement*

Machine learning algorithms demand a considerable amount of computing resources because they may handle large volumes of data and/or conduct heavy computations. As mentioned earlier, machine learning application development is an engineering task in which developers cut, add, modify, and tune the system so that the expected performance of the model can be achieved. Hence, there are some trial-and-test processes because a number of candidates are assessed for training and evaluated until a satisfactory model is identified. One of the major application domains of high-performance computing is machine learning applications. A private entity may not be able to afford to acquire sufficient computing resources for machine learning application development. Cloud services for machine learning application development are among the easiest types of enabling technology and are also among the most accessible computing systems.

4.1.9. *Algorithm implementation and public tool usage*

Once a machine learning algorithm is selected for use, it should be implemented as a code. The code can be implemented by a developer. Various excellent machine learning tools and frameworks have been developed for public use, and several machine learning tools are commercially available. Developers must determine which tools to use or to develop independently. This self-development requires considerable cost and time resources but provides much flexibility. Public and commercial tools have independent usage methodologies and a developer must therefore learn how to use them and to build the execution

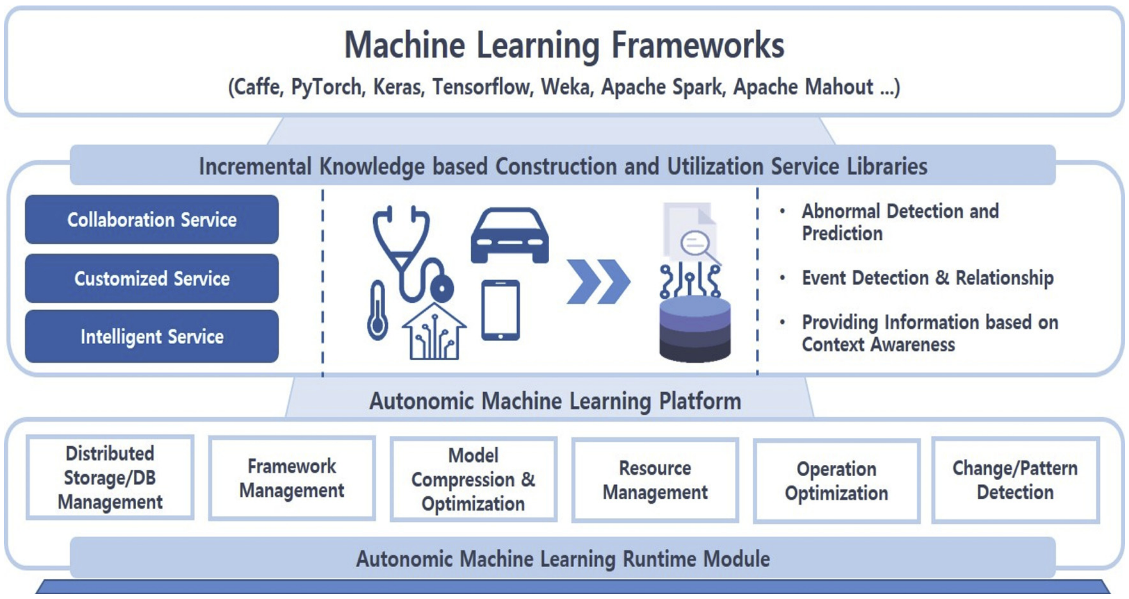

Machine Learning Frameworks
(Caffe, PyTorch, Keras, Tensorflow, Weka, Apache Spark, Apache Mahout ...)
Incremental Knowledge based Construction and Utilization Service Libraries
Collaboration Service
Customized Service
Intelligent Service
Abnormal Detection and Prediction
Event Detection & Relationship
Providing Information based on Context Awareness
Autonomic Machine Learning Platform
Distributed Storage/DB Management
Framework Management
Model Compression & Optimization
Resource Management
Operation Optimization
Change/Pattern Detection
Autonomic Machine Learning Runtime Module

Fig. 2. Functions of an Ideal Autonomic Machine Learning Platform.

environment with the employed tools.

4.1.10. *Execution of machine learning algorithms*

Machine learning algorithms take a certain amount of time to execute. During the development of a machine learning application, various machine learning models are usually assessed to find an excellent model. The parallel execution of candidate models is widely used during actual development processes. Developers must manage the deployment and monitoring of the model training tasks over the computing resources. When a model seems destined to fail, it is better to terminate the execution of the algorithm.

4.1.11. *Evaluations of trained models*

Multiple machine learning models are trained with different configurations and hyperparameters. To select the most effective ones, evaluations must be done with respect to certain evaluation criteria and test data. Developers need to prepare a procedure and/or schedule for such machine learning model evaluations.

4.1.12. *Model updates for data distribution changes in the problem domain*

Occasionally, data distribution in a problem domain may change over time. As time passes, more data are collected. Machine learning models must be improved as more data are collected and used for learning. Developers therefore need to undertake actions to improve the model as more data are collected.

Even when an expert is devoted to developing a machine learning application with due consideration of the above-mentioned design factors, it also takes considerable computing resources for multiple trials to find a model capable of satisfactory performance levels. Hence, it will be very useful to have a platform to help machine learning applications without requiring a machine learning expert and to manage the computing resources required to search for the best model and its hyperparameters properly.

4.2. *Requirements of an autonomic machine learning platform*

An autonomic machine learning platform aims to provide an environment in which a non-expert in the area of machine learning develops an application based on machine learning for the problem at hand. There are several requirements for such a machine learning platform:

First, the platform must provide a mechanism such that existing public machine learning frameworks and tools can be used. There are excellent machine learning tools, some of which are even open-source software supported by major companies or well-known open-source software communities. Representative machine learning tools are SparkML, TensorFlow, Apache Mahout, and Weka. There are also several language environments, such as Apache Spark and Python, which allow massive data processing and complicated computations. It is important to allow developers to use familiar tools when they face a new development environment. Various well-known and rapidly developing tools are efficient and effective for machine learning experts to use. An autonomic machine learning platform hence supports such tools in its environment.

Second, the platform should enable non-experts to build machine learning applications even when they do not have sufficient understanding of machine learning algorithms or the skills to implement their complicated logic. We define the autonomic level at that in which the system handles machine learning tasks in an autonomic manner. We posit five autonomic levels.

Third, the platform must have the capability of using external computing resources and executing learning and inference tasks with them. The platform should have mechanisms for registering, locating, and monitoring the machine learning computing resources and machine learning tools and for assigning machine learning tasks to them.

Fourth, the platform must provide a script language with which the autonomic machine learning tasks are expressed in a distributed and parallel manner. Script languages can be either coarse-grained or fine-grained. Coarse-grained script languages allow programmers to express tasks and processes in a fairly abstract manner. Fine-grained script languages allow them to specify the details of the tasks and processes. Some existing scripting language tools are equipped with a fine-grained script language in which developers design their script language models and implement the logic of the script language algorithms. Such script languages at times express the script language models and the operations of script language algorithms in directed graphs, such as a data flow graph in TensorFlow and an RDD lineage graph in Apache Spark. Fine-grained script languages are tightly bound to the execution platform on which they are compiled, deployed, and executed. Coarse-grained script languages deal with functional modules that carry out certain designated tasks and express the process of operations with the functional modules. Once machine learning applications are written in a coarse-grained script language, it is more convenient to execute them on a distributed and parallel environment because scheduling and management can be done on the basis of a functional module.

Fig. 2 shows all of the functions required by an ideal autonomic machine learning platform. An ideal autonomic machine learning platform should provide several subsystems as well, including those for distributed storage management, framework management, model compression, model optimization, resource management, operating optimization, and change detection. Distributed storage and resource management subsystems handle principally big data. The operation optimization subsystem provides real-time operations in conjunction with real-time data to make many real-world decisions. The change detection subsystem builds a general description for considering changes in the input data while the pattern detection subsystem serves to classify data based on knowledge already gained or on statistical information extracted from patterns. The model compression subsystem provides model acceleration through both machine algorithm-level and system-level optimization.

The autonomic machine learning platform also includes a set of core service libraries that provide the functionality of incremental knowledge for simplifying and accelerating the training and construction of various machine learning models by means of a collaboration service, customized service, and intelligent service. These services should detect abnormal results, various events, and the relationships between events based on context awareness.

5. Autonomic machine learning platform

5.1. Architecture of the proposed autonomic machine learning platform

To implement the proposed autonomic machine learning platform, we design an architecture that meets the abovementioned requirements, as shown in Fig. 3. The architecture supports both black box and white box modes.

In the black box mode, the machine learning platform does not ask users to write codes of machine learning tasks but simply asks them to specify the components corresponding to the autonomic level employed at the moment. The remaining details of machine learning tasks are handled by the platform, which uses existing machine learning platforms and integrated tools. In the black box approach, the platform contains various functional modules for both the machine learning algorithms and design factors mentioned in Section 4.1.

Users of an autonomic machine learning platform do not have to be strongly engaged in machine learning model development, with the exception being the design factors required for experts to provide. Hence, users do not have to undertake fine control of the development of the machine learning application in terms of machine learning model construction and/or computing resource utilization.

In the white box mode, users write codes for machine learning tasks according to their own logic and design choices using the provided script programming language. The machine learning models and the algorithms for training and inference are encoded in a directed computation graph which is later compiled and executed. The deployment of operations in the directed computation graph can be more complicated than coarse-grained scripts because the granularity of the operations may not be large enough or the data communication between operations is heavy. When operations are tightly bound, it is better to wrap them into a module which runs a single computer system.

Users can also interact with the platform through ML (machine learning) clients. ML clients deliver user requests to the autonomic ML coordinator, monitor the progress of the deployed tasks, and receive the learning and inference results. This component maintains the connection between the database system and the platform so that the data sets in the database can be used during machine learning tasks, with the results of learning and inference stored and maintained in a database. The autonomic ML coordinator takes charge of coordinating autonomic machine learning tasks by generating machine learning task plans for plausible configurations, executing them with the available computing platforms, and selecting the best models and patterns from the results of the machine learning tasks. Due to the enormous number of candidate machine learning configurations, the demands on computing resources can soar as the autonomic level becomes higher. In such a case, expert knowledge can be helpful to establish the initial configurations and to narrow down the ranges of the candidate hyperparameters under consideration.

Such knowledge is maintained in the ML plan repository, into which execution profiles, including the configurations and their performance outcomes, are maintained to enhance knowledge incrementally. The ML plan repository also includes a cost model for the configurations and data characteristics. The cost model is updated to reflect the execution profiles collected during the machine learning tasks conducted on the platform. The ML planner is in charge of determining the model

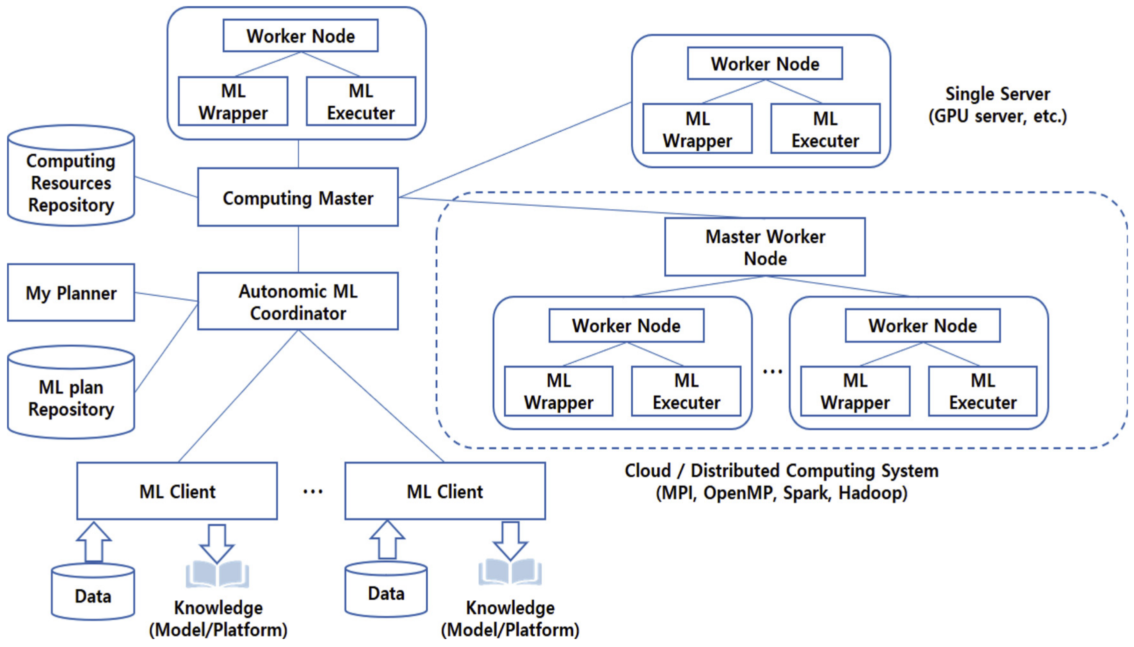


Fig. 3. Architecture of the proposed autonomic machine learning platform.

configuration and hyperparameters, and it undertakes feature selection and extraction as well as machine learning task deployment.

The computing master receives machine learning tasks from the autonomic ML coordinator, deploys them over the computing resources, monitors their progress, reconfigures the deployments of the machine learning tasks, if needed, and reports the status and the results to the autonomic ML coordinator. The computing resources include single servers such as GPU servers and cloud/distributed computing systems. Each computing resource has a master worker node which communicates with the computing master and manages its own computing resources to conduct machine learning tasks. The proposed platform supports the use of existing machine learning tools and frameworks. Hence, the machine learning wrappers are implemented to invoke machine learning tools to execute the corresponding machine learning tasks, to report their progress and results, and to terminate the deployed tasks if needed.

When the machine learning tasks are expressed in the script language, the codes are executed by the ML executer, which launches the corresponding machine learning platform and conducts the execution of the codes on the platform according to the corresponding configuration. When a cloud/distributed computing system is used as a computing resource, the master worker node is installed on it.

The master worker node communicates with the deployed machine learning platform or frameworks which handle machine learning tasks. If the machine learning codes in the script language are delivered, the master worker node distributes the machine learning codes to the worker nodes, which then invoke the ML executors. The master worker nodes also support their registering into the computing resource repository with their access methods, functionalities, capabilities, workloads, and billing policies. The computing resource repository checks the registered computing resources on a regular basis and maintains the current states. The computing master refers to the computing resource repository when distributing machine learning tasks and monitoring their progress.

5.2. Design of the proposed machine learning platform

The proposed autonomic machine learning platform is a very large system which integrates existing machine learning tools and frameworks to use many external computing resources. In addition, it is an ambitious approach to attempt to realize a full range of autonomic levels, from the primitive level to the completely autonomic level. Under the black box approach and with coarse-grained script support, the architecture and the functional modules are designed and specified.

The proposed autonomic machine learning platform consists of a library that implements algorithms by software with modularization and interfaces that invoke the configuration modules of such libraries and use them as a tool, as shown in Table 2.

The functional specifications of the components of the proposed autonomic machine learning platform are as follows.

5.2.1. User manager

It provides registration/modification/withdrawal functions for the users of the autonomic machine learning framework and works with a user login module.

5.2.2. Project manager

All machine learning activities are performed on a project-by-user basis, with history management also performed. The project management module registers the project code (identifier), the project name, the task type, the task category, and the project description.

5.2.3. Resource manager

When the user is registered, this component automatically creates a file structure on the server or in a local repository and prepares to store the metadata and raw files of the data. Then, when the project is created, the actual data information is extracted, the summary information is stored in a database, and the actual data file is stored in a pre-defined location.

5.2.4. Data processor

This handles information about data processor registrations/modifications/deletions of project code, data management numbers (identifiers), data names, data separation, data sizes, batch sizes, data shuffles (true/false), and file paths. When the data processing is completed, a setting script for the machine learning data is generated.

5.2.5. Model constructor

A model code, project code, machine learning algorithm code, layer code, and sequence code are registered to construct a model. Machine learning algorithms register algorithm codes, algorithm names, and descriptions. The layer module registers the layer code, the layer name, and the layer description as information for the layer configuration of the neural network when the deep learning model is constructed. The sequence module registers the sequence information for the configuration of the neural network.

5.2.6. Wrapper manager

Once the model configuration is complete, the user imports a package of related machine learning frameworks, such as TensorFlow, Keras, Route-run or Pie-Pie. A wrapper script is created by adding a machine learning data input/output label setting script and the machine learning model information generated by the data processor.

5.2.7. Model manager

This component systematically stores and manages metadata

Table 2
Autonomic machine learning component function classification.

Function	Detail of Function
User Management	User Management Module
Project Management	Project Management Module: Manage all machine learning courses on a project basis
Data Processing	Systematically store and manage data information files and unprocessed data necessary for machine learning Set up data I/O for learning and generate Python scripts
Model Construction	Generate and manage scripts for data modeling and learning modeling
Wrapper Management	Mange Machine Learning Model Script Information Generated By AUTO ML
Model Management	AUTO ML model management enables efficient deployment and management of Machine Learning models built using a variety of frameworks, including SparkML, Keras, TensorFlow, or Python
Model Executor	Configuring the Model Execution Environment
Resource Management	Location information management and computing resource management of machine learning algorithm
Execution Management	Automatically captures information about each run for each model and provides a complete record of the project (experiment) in the form of run history
Code Management	Common Code (Identifier) Management

pertaining to the user who performed machine learning and the data pertaining to the wrapper. It also automatically captures information about each run for all models and records and manages these data on a project-by-project basis.

5.2.8. Model executor

In order to train a learning model in a multi-parallel environment, information about the environment description and execution commands is written in a script and stored and managed so that it can be reused when needed.

In this paper, we do not consider performing machine learning through an adjustment to various hyperparameters. The proposed autonomic machine learning platform is a technology that automatically acquires meaningful knowledge while minimizing expert intervention by allowing machine learning to be performed at various autonomic levels. In addition, the proposed platform can be easily extended to various functions because existing machine learning algorithms can be used as a framework for providing common APIs (application programming interfaces) and can be embedded in platforms that can be used in various system environments.

5.3. Application to smart cities and economical implication

In this section, we demonstrate that the proposed autonomic machine learning platform is especially suitable for a smart city, where large-scale machine learning applications are applied. We also explain the economic effects when the proposed autonomic machine learning platform is applied to a smart city. Fig. 4 shows the applications of autonomic machine learning for a smart city, smart factory, and a smart grid.

In a smart city, smart factory, and smart grid, a large volume of data must be captured, aggregated, smoothed, clustered, and processed. The proposed autonomic machine learning platform can be applied to clustering and knowledge extraction given that the existing database engine is capable of statistical analysis, allowing the user to utilize this function to perform analyses or to execute specific functions in the form of a trigger.

Information detection, information prediction, prognosis information analyses, and relationship analyses are typical services in a smart city, smart factory, and smart grid. In order to provide these services through existing machine learning tools, experts must check the characteristics of the data again and, in the event of a major change, must learn the new machine learning model by using it. However, the proposed autonomic learning platform can detect changes in the very large databases compiled by smart cities, smart factories, and smart grids without expert intervention and can automatically extract new knowledge and provide various services.

One problem that can arise when providing various services in smart city, smart factory, and smart grid applications is that these applications contain typical structured data as well as unstructured big data. In order to analyze big data given its various characteristics using existing machine learning tools, it may be necessary to hire other experts depending on the characteristics of the data, and it can be very expensive to obtain meaningful information and provide various services as well. However, because the proposed autonomic learning platform does not require expert intervention, it can detect changes in large huge databases without an additional cost and can automatically extract new knowledge and provide various services.

Due to the relevance of smart cities to various stakeholders and the benefits and challenges associated with its implementation, Smart cities employ information and communication technologies to improve (Ismagilova, Hughes, Dwivedi, & Raman, 2019). In particular, smart

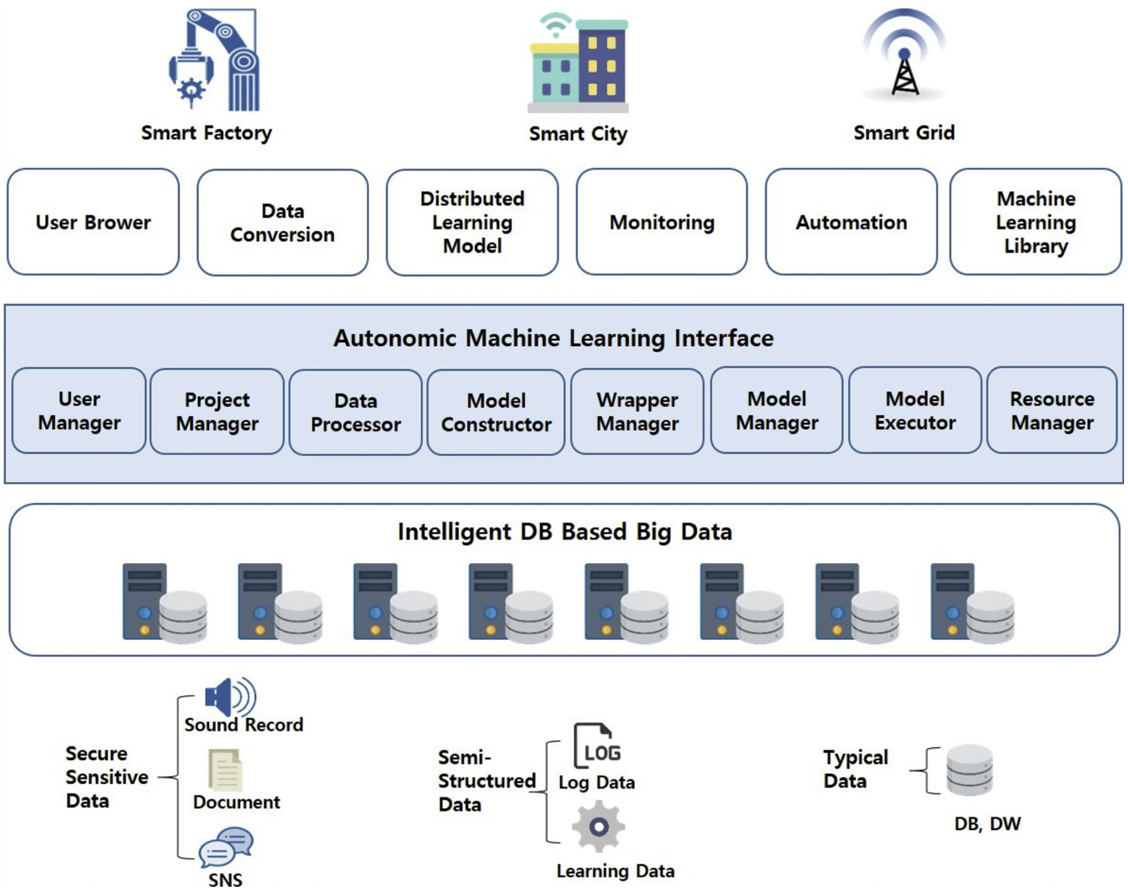


Fig. 4. Autonomic machine learning Structure for Smart Cities, Factories, and Grids.

city applications have considerable amounts of user-sensitive data compared to other applications, and collecting meaningful information using them can cause privacy problems. Therefore, it is important to develop an efficient IT infrastructure which must focus on security issues (Rana et al., 2018). Otherwise, smart city applications require additional security systems, which in turn will incur significant additional costs. However, the proposed autonomic learning platform is the best solution for machine learning for smart city applications because it can minimize the need for expert intervention by performing machine learning at various autonomic levels according to the sensitivity of the data without additional security systems. In addition, if the proposed autonomic machine learning platform is fully implemented, in the short term technological underuse or structural underuse can be avoided. On the other hand, in the long term, the proposed autonomic machine learning platform is more economical because it can extract meaningful data without additional costs while also protecting sensitive data.

6. Discussion

This study aims to present the need of the autonomic machine learning platform for the universal use of machine learning techniques in a variety of applications including Smart City, Smart Factory, and Smart Grid.

This study has several unique contributions and implications, given as follow:

- This study presents twelve design factors to be required by expert knowledge and intuition during the machine learning development process.
- This study defines five levels of autonomic machine learning referring to as the degree of expert interventions based on the steps of the machine learning development process.
- The levels of autonomic machine learning can minimize expert intervention at various autonomic levels by reducing the number of design selections that require expert knowledge and intuition is proposed. This autonomic machine learning platform can be used to derive a high-quality learning result.
- This study focuses on the design issues in terms of the practical autonomic machine learning by applying the autonomic machine learning related to smart cities from an information systems perspective. Therefore, this study is useful for system developers involved in smart city development initiatives using machine learning.
- In a truly smart city of the future, automation will be paramount to improve the service level of the end users (Rana et al., 2018). This capability can be derived from advanced information technologies such as the proposed autonomic machine learning platform.

Like any publication, this study has certain limitations, given as follow:

- This study only focuses the design of the autonomous machine learning platform, but the actual implementation or application may be very different from the proposed design structure and it does not cover issues related to implementing the autonomic machine learning techniques and systems.
- As a research study, no primary data was collected or used to support the development of the proposed autonomic machine learning platform.
- This study could not provide a valuable synthesis of the relevant literature by analyzing and discussing the key findings from the existing machine learning platforms on issues related to smart cities from an Information Systems Perspective.

7. Conclusions

In this paper, we proposed an autonomic machine learning platform by defining autonomic levels from the primitive level to the ultimate level. We also designed a type of architecture for the realization of an autonomic machine learning platform which uses external computing resources and existing machine learning platforms and frameworks. The architecture handles the vast demands on computing resources typically found in autonomic machine learning. We also showed that the proposed autonomic machine learning is appropriate for a smart city, which typically has significant amounts of sensitive data and various IoT (Internet of Things) devices.

The proposed autonomic machine learning platform can be an important enabling technique to convert a database into an intelligent database. With the advances and widespread deployment of sensor technologies such as IoT devices, large volumes of data have been accumulated, as expected. It is, however, not easy to transform such data into valuable knowledge. The proposed autonomic machine learning platform can help to extract valuable knowledge at high confidence levels from big data.

There yet remains much work before the functionalities of the autonomic machine learning platform can be fully realized. However, once it is implemented, machine learning technology can be easily adopted in various practical business domains without machine learning expert intervention. Specifically, databases can evolve into intelligent databases with the help of the proposed autonomic machine learning platform. Such intelligent databases allow the dream that once data are stored, knowledge is developed automatically.

Acknowledgements

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, and ICT (No.NRF-2017M3C4A7069432).

References

Autonomic Computing Strategy Perspectives (2018). *Autonomic computing strategy perspectives.* Retrieved fromhttp://ptgmedia.pearsoncmg.com/images/0131440241/samplechapter/0131440241_ch03.pdf.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Proceedings of the 2011 advances in neural information processing systems,* 2546–2554.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13,* 281–305.

Branscombe, M. (2018). *9 Misunderstandings about machine learning.* Retrieved fromhttp://www.ciokorea.com/news/37676#csidx531708c30dfe3e09ee899143dab28e4/.

Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR, abs/1012.2599* https://www.semanticscholar.org/paper/A-Tutorial-on-Bayesian-Optimization-of-Expensive-to-Brochu-Cora/cd5a26b89f0799db1cbc1dff5607cb6815739fe7.

Cai, Z., Gao, Z. J., Luo, S., Perez, L. L., Vagena, Z., & Jermaine, C. (2014). A comparison of platforms for implementing and running very large scale machine learning algorithms. *Proceedings of the 2014 ACM SIGMOD international conference on management of data,* 1371–1382.

Chandrasegaran, K. (2017). *The non-techie's guide to machine learning.* Retrieved fromhttps://medium.com/swlh/the-non-techies-guide-to-machine-learning-45337265e01e/.

Duan, Y., Edwards, J. S., & Dwivedi, Y. K. (2019). Artificial intelligence for decision making in the era of Big Data–evolution, challenges and research agenda. *International Journal of Information Management, 48,* 63–71.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Proceedings of the 2015 advances in neural information processing systems,* 2962–2970.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., & Sculley, D. (2017). Google vizier: A service for black-box optimization. *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining,* 1487–1495.

Google (2018). *Cloud AutoML BETA.* Retrieved fromhttps://cloud.google.com/automl/.

Ismagilova, E., Hughes, L., Dwivedi, Y. K., & Raman, K. R. (2019). Smart cities: Advances in research—An information systems perspective. *International Journal of Information Management, 47,* 88–100.

Kang, S. J., Lee, S. Y., & Lee, K. M. (2015). Performance comparison of OpenMP, MPI, and MapReduce in practical problems. *Advances in Multimedia, 2015,* 1–9.

Kotthoff, L. (2018). *Auto-WEKA.* Retrieved fromhttps://www.cs.ubc.ca/labs/beta/Projects/autoweka/.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2017). An empirical evaluation of deep architectures on problems with many factors of variation. *Proceedings of the 24th international conference on machine learning,* 473–480.

Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., & Ramchandran, K. (2018). Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory, 64*(3), 1514–1529.

ML Freiburg (2018). *SMAC*. Retrieved fromhttps://www.ml4aad.org/automated-algorithm-design/algorithm-configuration/smac/.

Moore, J. H. (2018). *Information about automated machine learning (AutoML)*. Retrieved fromhttps://automl.info/.

Olson, R. S., & Moore, J. H. (2016). TPOT: A tree-based pipeline optimization tool for automating machine learning. *Proceedings of the 2016 workshop on automatic machine learning,* 66–74.

Rana, N. P., Luthra, S., Mangla, S. K., Islam, R., Roderick, S., & Dwivedi, Y. K. (2018). Barriers to the development of smart cities in Indian context. *Information Systems Frontiers,* 1–23. https://doi.org/10.1007/s10796-018-9873-4.

Sparks, E. R., Talwalkar, A., Smith, V., Kottalam, J., Pan, X., Gonzalez, J., et al. (2013). MLI: An API for distributed machine learning. *Proceedings of the 2013 IEEE 13th international conference on data mining,* 1187–1192.

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining,* 847–855.

Keon Myung Lee is a professor in the Department of Computer Science, Chungbuk National University, Korea. He received his B.S., M.S., and Ph.D. degrees in computer science from KAIST, Korea and was a post-doctorate fellow at INSA de Lyon, France. He worked for Park Scientific Instrument, CA, USA. He was a visiting professor at the University of Colorado at Denver and a visiting scholar at Indiana University, USA. His principal research interests are Data Mining, Machine Learning, Soft Computing, Big Data Processing, and Intelligent Service Systems.

Jaesoo Yoo received Ph.D. in Computer Science from Korea Advanced Institute of Science and Technology(KAIST), Korea in 1995. He is now a professor in School of Information and Communication Engineering, Chungbuk National University, Korea. His research interests are Big Data Processing, Distributed Computing, Machine Learning, and Social Network Services.

Sang-Wook Kim received the BS degree in Computer Engineering from Seoul National University, Korea in 1989 and earned the MS and PhD degrees from Korea Advanced Science and Technology (KAIST) at 1991 and 1994, respectively. He is a Professor at Department of Computer Science and Engineering, Hanyang University, Korea. Professor Kim worked with Carnegie Mellon University and IBM Watson Research Center as a visiting scholar. He is an associate editor of Information Sciences. His research interests include Data Mining and Databases.

Jee-Hyong Lee received his B.S., M.S., and Ph.D. in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1993, 1995, and 1999, respectively. From 2000 to 2002. He was an international fellow at SRI International, USA. He joined Sungkyunkwan University, Suwon, Korea, as a faculty member in 2002. His research interests include Machine Learning, Deep Learning, Data Mining and Text mining.

Jiman Hong received B.S degree in Computer Science from Korea University, Seoul, Korea and his M.E. and Ph.D. degrees in Computer Science and Engineering from Seoul National University. He has been with School of Computer Science and Engineering, Soongsil University, Seoul, Korea since 2007, where he is a full professor. Previously he was the assistant professor at the Department of Computer Engineering, Kwangwoon University, Seoul, Korea between March 2004 and February 2007. He worked as a Chief of Technical Officer in the R&D center of GmanTech Incorporated Company, Seoul Korea between March 2000 and December 2003. His research interests include Operating Systems, Machine Learning Software Platforms, and IoT Systems. He has served as technical and organizational committees, program chair, poster chair, workshop chair of more than 40 IEEE and ACM international conferences. He is currently serving as the Chair of ACM SIGAPP.